OIPE JC101
AUG 2 2 2005
PATENT & TRADEMARK OFFICE

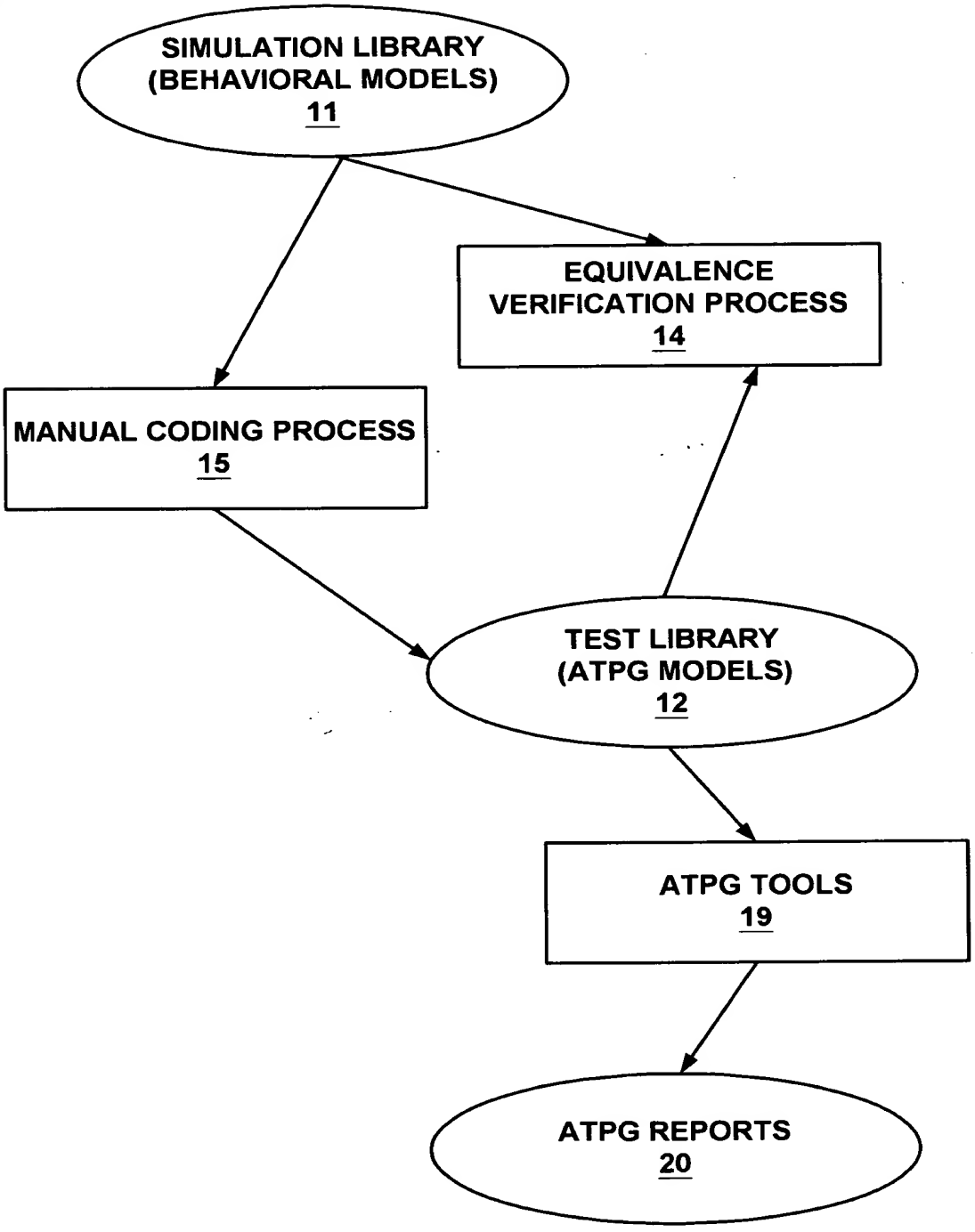

FIGURE 1
(PRIOR ART)

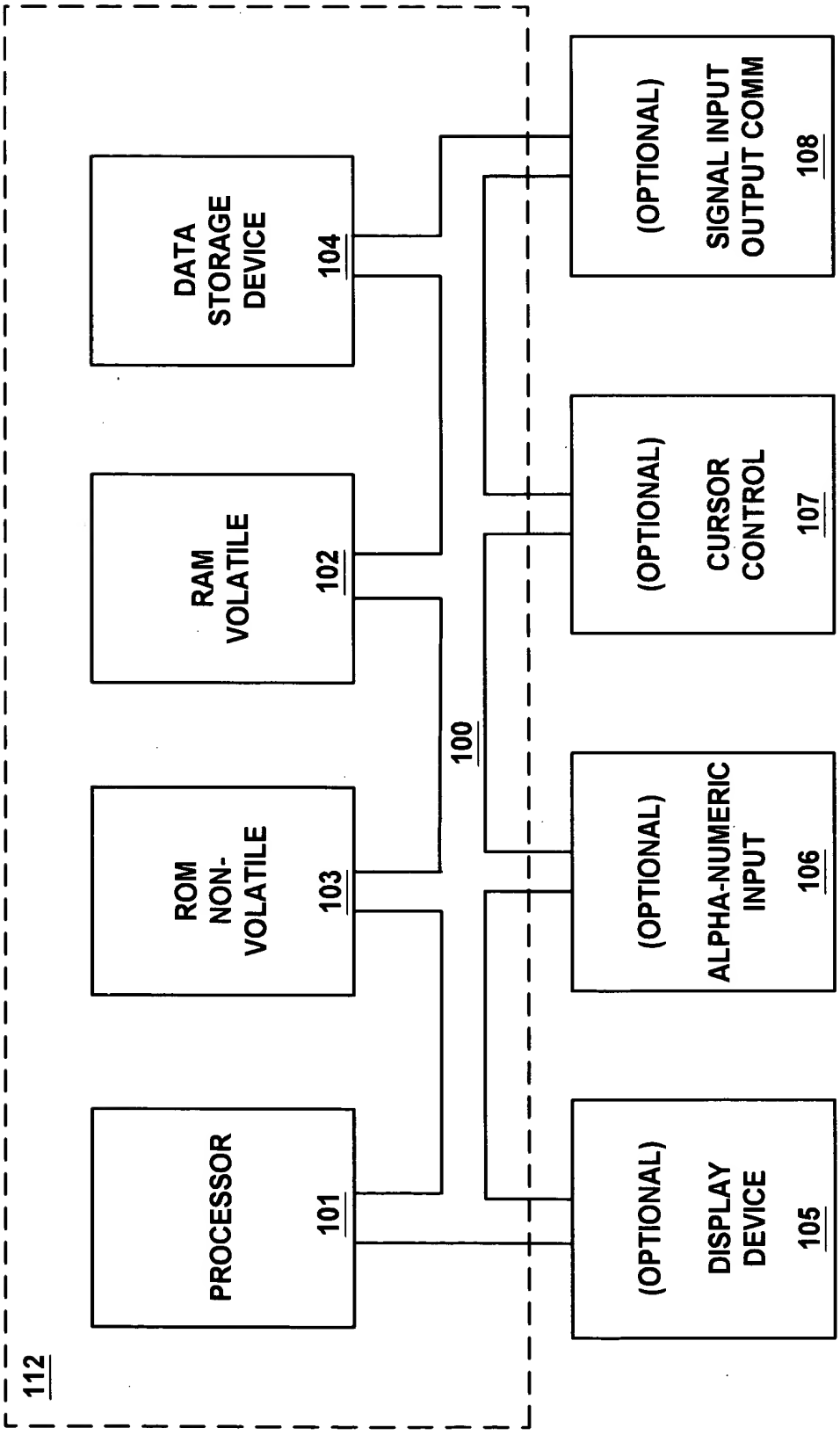
112
PROCESSOR
101
ROM
NON-
VOLATILE
103
RAM
VOLATILE
102
DATA
STORAGE
DEVICE
104
100
(OPTIONAL)
DISPLAY
DEVICE
105
(OPTIONAL)
ALPHA-NUMERIC
INPUT
106
(OPTIONAL)
CURSOR
CONTROL
107
(OPTIONAL)
SIGNAL INPUT
OUTPUT COMM
108

FIGURE 2

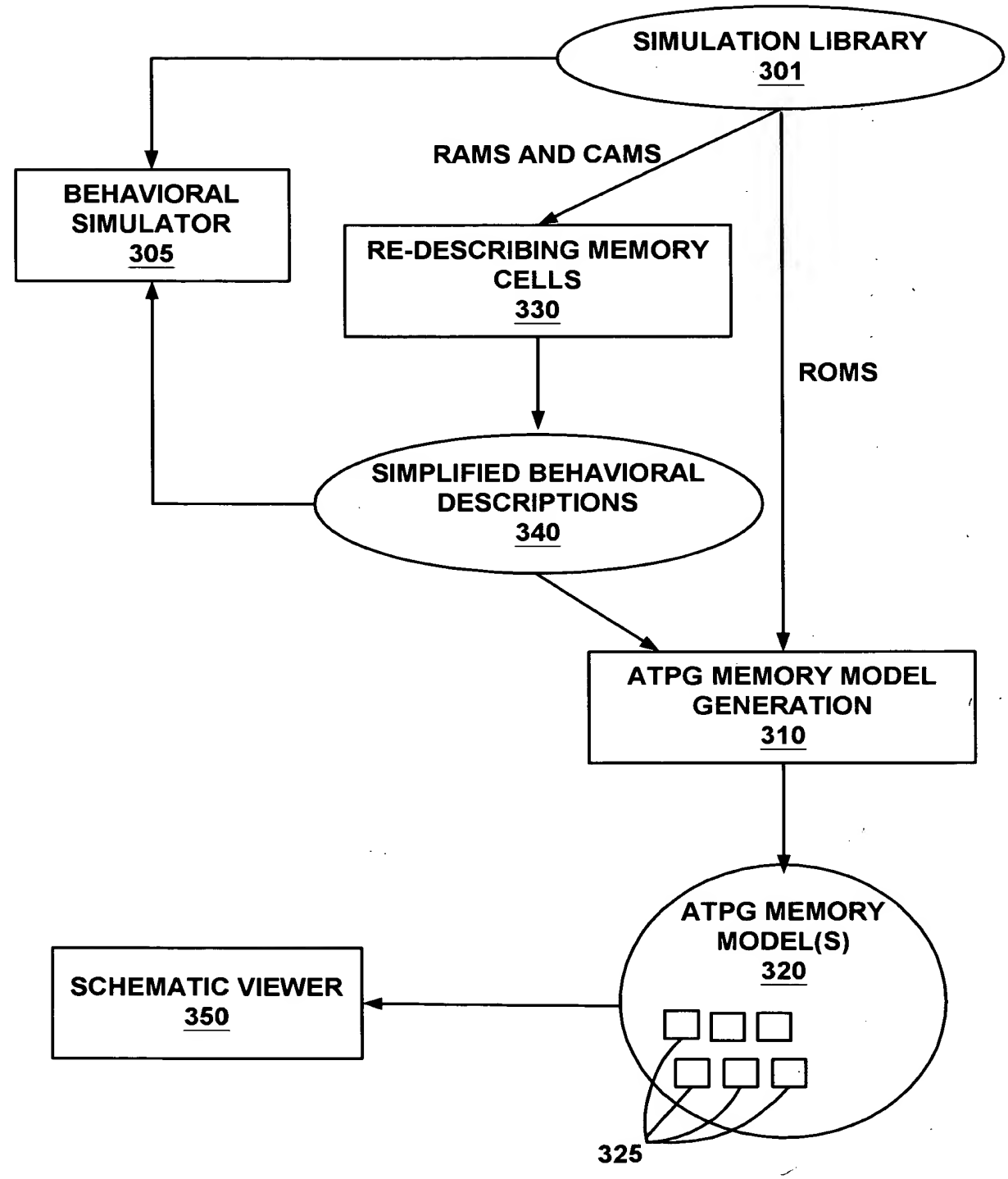
300
SIMULATION LIBRARY
301
RAMS AND CAMS
BEHAVIORAL SIMULATOR
305
RE-DESCRIBING MEMORY CELLS
330
ROMS
SIMPLIFIED BEHAVIORAL DESCRIPTIONS
340
ATPG MEMORY MODEL GENERATION
310
ATPG MEMORY MODEL(S)
320
SCHEMATIC VIEWER
350
325

FIGURE 3

400

```
memory_port : : = read_port I write_port I set_port I reset_port
read_port : : = always @ ( level_read_port I edge_read_port )
write_port : : = always @ ( level_write_port I edge_write_port )
reset_port : : = [begin] sr_control memory_name [address_net] <= 0_constant; [signal]
set_port : : = [begin] sr_control memory_name [address_net] <= 1_constant; [signal]
level_read_port : : = [ ( ] address_net [or control_net] [or event] [ ) ]
    (net_control I addr_control) read_port_assign [ else reg_net <= constant ; ]
level_write_port : : = ( address_net or control_net or data_net )
    net_control write_port_assign
edge_read_port : : = (edge clock_net) (net_control I addr_control) read_port_assign
edge_write_port : : = (edge clock_net) [net_control] write_port_assign
read_port_assign : : = reg_net <= memory_name [ address_net ] ;
write_port_assign : : = [begin] memory_name [ address_net ] <= data_net ; [signal]
sr_control : : = always @ control_net net_control
          for ( index = 0 ; index < max_address; index = index + 1 )
net_control : : = if ( [!] control_net )
addr_control : : = if ( address_net <= max_address )
signal :: = #0; -> event ; end
constant : : = number_bits ` base { digit } +
edge : : = posedge I negedge
base : : = b I B I h I H
digit : : = 0 I 1 I 2 I 3 I 4 I 5 I 6 I 7 I 8 I 9 I 0 I a I A I b I B I c I C I d I D I e I E I f I F I x I X z I Z
```

FIGURE 4

500

```
`celldefine
`ifdef verifault
    `suppress_faults
    `enable_portfaults
`endif
module ROM ( ... ) ;
output ...; input ...; wire ...;
reg [3:0] _doutr;
reg [3:0] rom_data [ 0:127 ] ;
    // control circuit
    buf ( _H15, H15 ) ;
    ....
    //data out
    buf ( N01,_dout [0] );
    ....
    //address
    buf ( _H07, H07 ) ;
    buf ( _H14; H14 ) ;
    and ( _a [6], _H07, _H15 );
    and ( _ta [6], _H14, _TEB ) ;
    or ( _ad [6], _a [6], _ta [6] );
initial
        $readmemh ("file" , rom_data );
always @ _ad // read port
        if ( _ad   <= 127 )
            _doutr = rom_data [_ad];
        else
            _doutr = 4'bx;
specify
...
endspecify
endmodule
`ifdef verifault
    `nosupress_faults
    `disable_portfaults
`endif
`endcelldefine
```

FIGURE 5
(CONVENTIONAL ART)

600

```
module withram (reset,  r,w,  a,  d1, d2 ) ;
input reset,  r,w;
input  [3:0]  a;
input  [7:0]  d1;
output  [7:0]  d2;
    reg  [7:0]  mymem  [15:0],  d2;
    integer  i;
    always  @  reset if  (reset)
    for  (i=0;  i<16;  i = i + 1)  mymem [i]  <= 0;
    always  @  (posedge w)  mymem [a] <=  d1;
    always  @  r if  (r)  d2  <=  mymem [a];
        else  d2  <= 0;  //  read_off is 0
endmodule
```

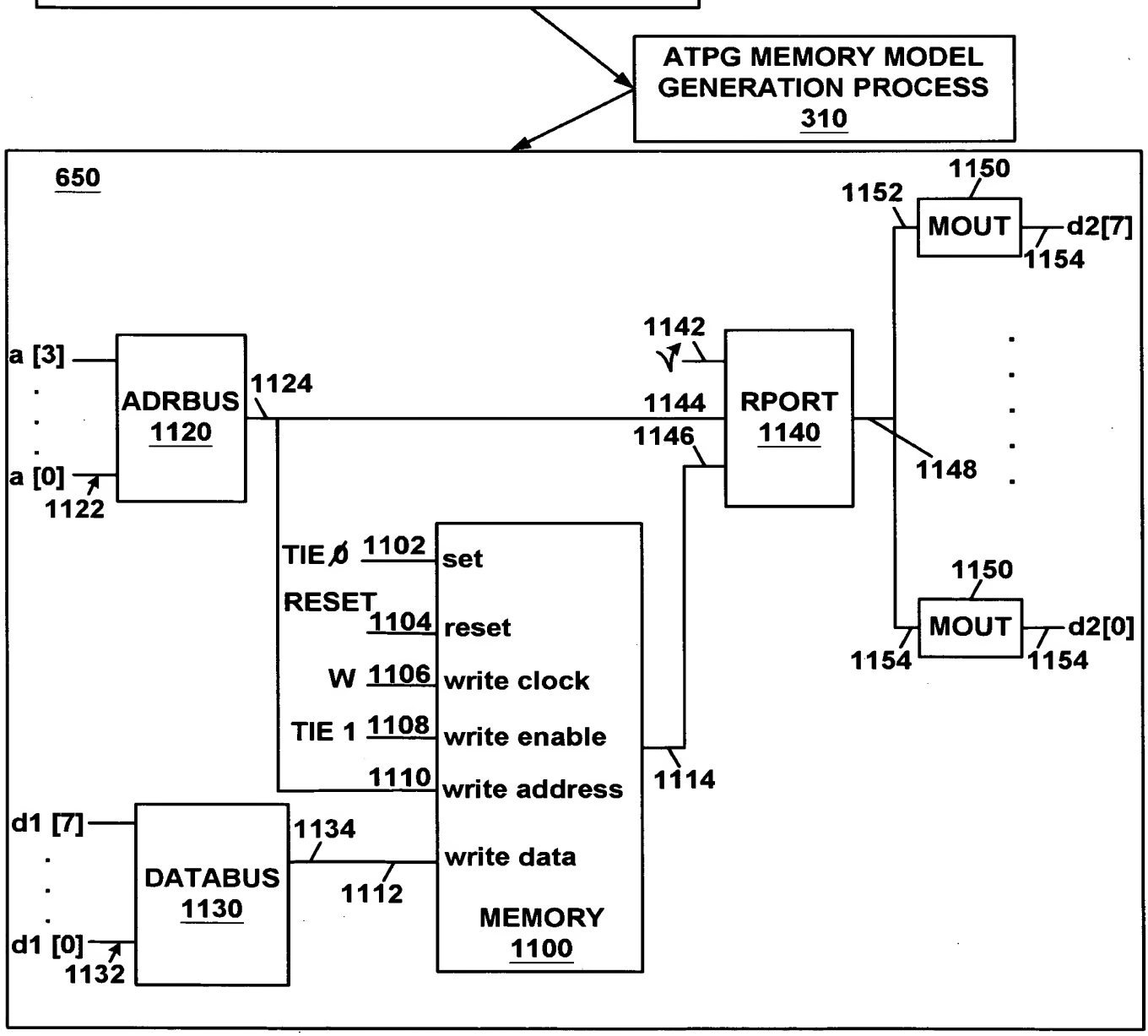

FIGURE 6A

310

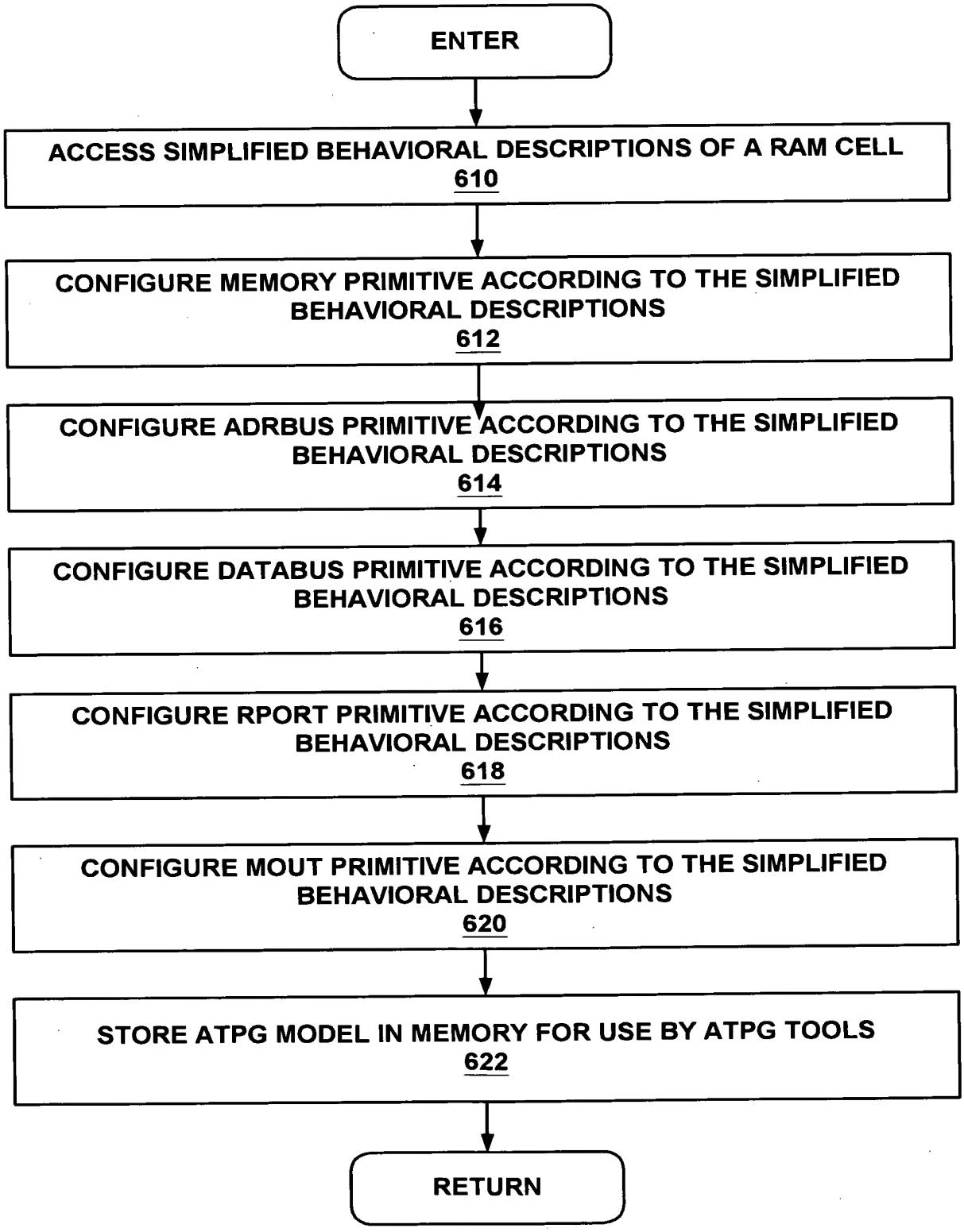

FIGURE 6B

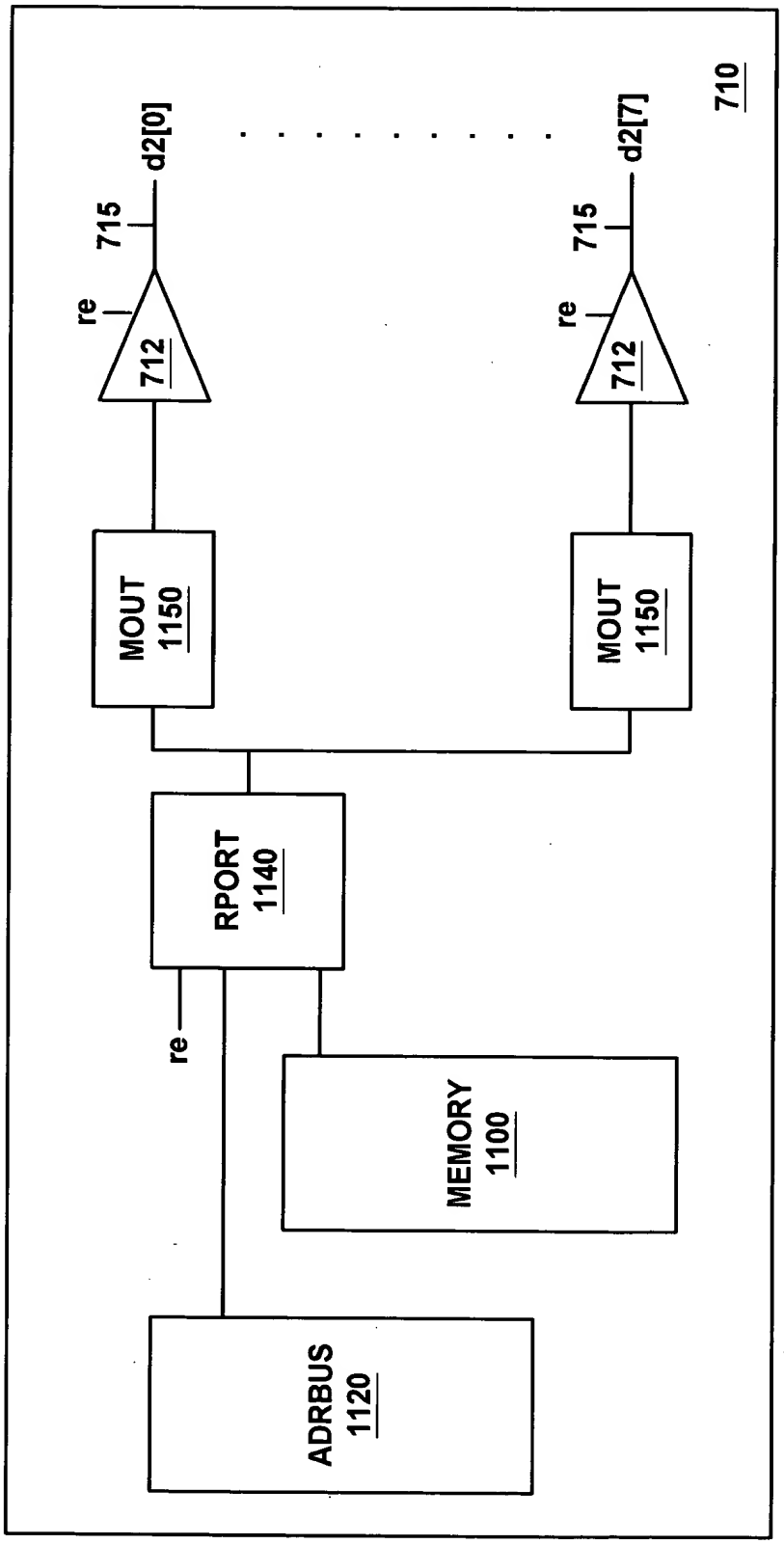
710
d2[0]
d2[7]
715
715
re
re
712
712
MOUT
1150
MOUT
1150
RPORT
1140
re
MEMORY
1100
ADRBUS
1120

FIGURE 7A

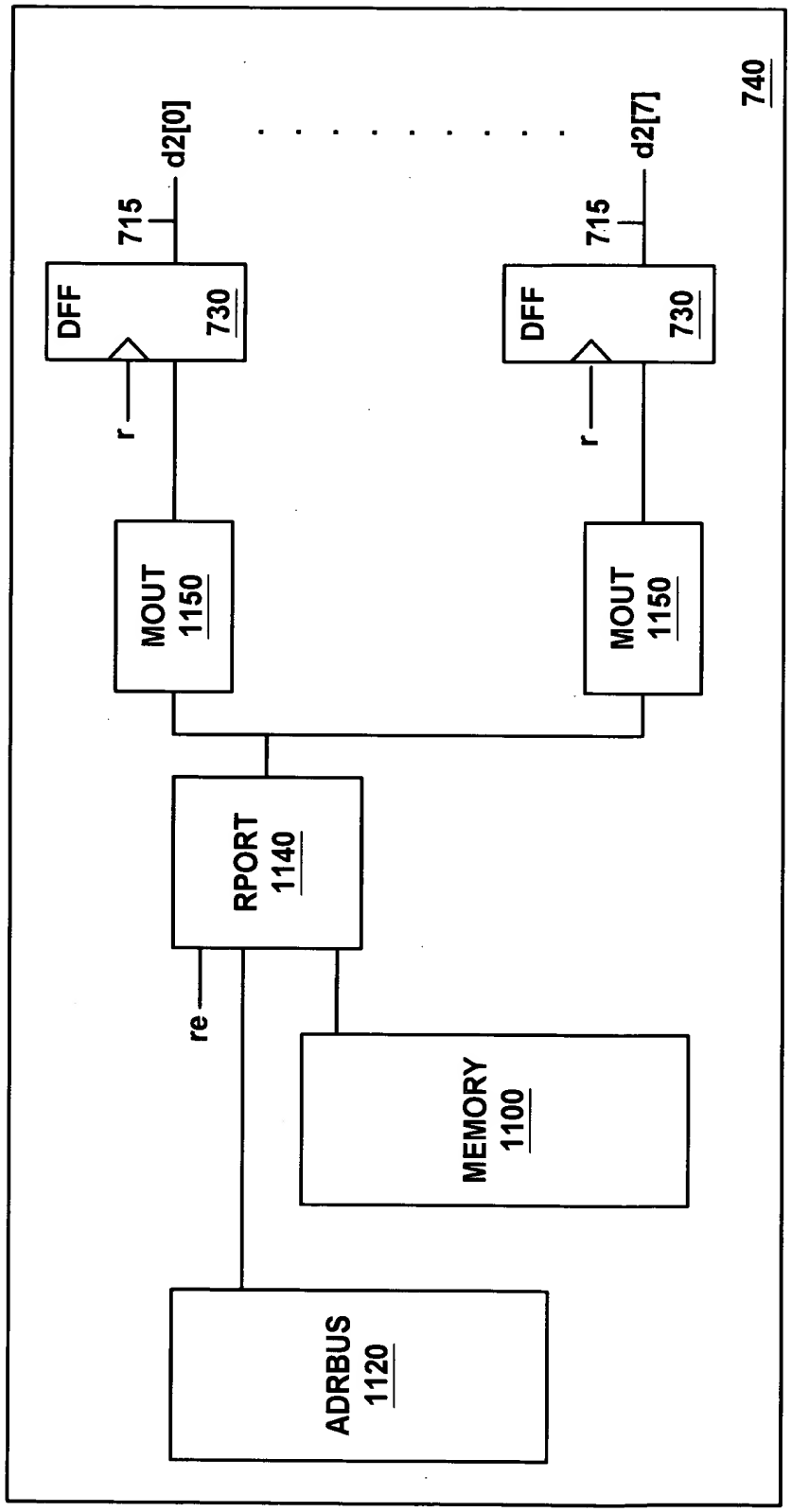
740
d2[0]
d2[7]
715
715
DFF
730
DFF
730
r
r
MOUT
1150
MOUT
1150
RPORT
1140
re
MEMORY
1100
ADRBUS
1120

FIGURE 7B

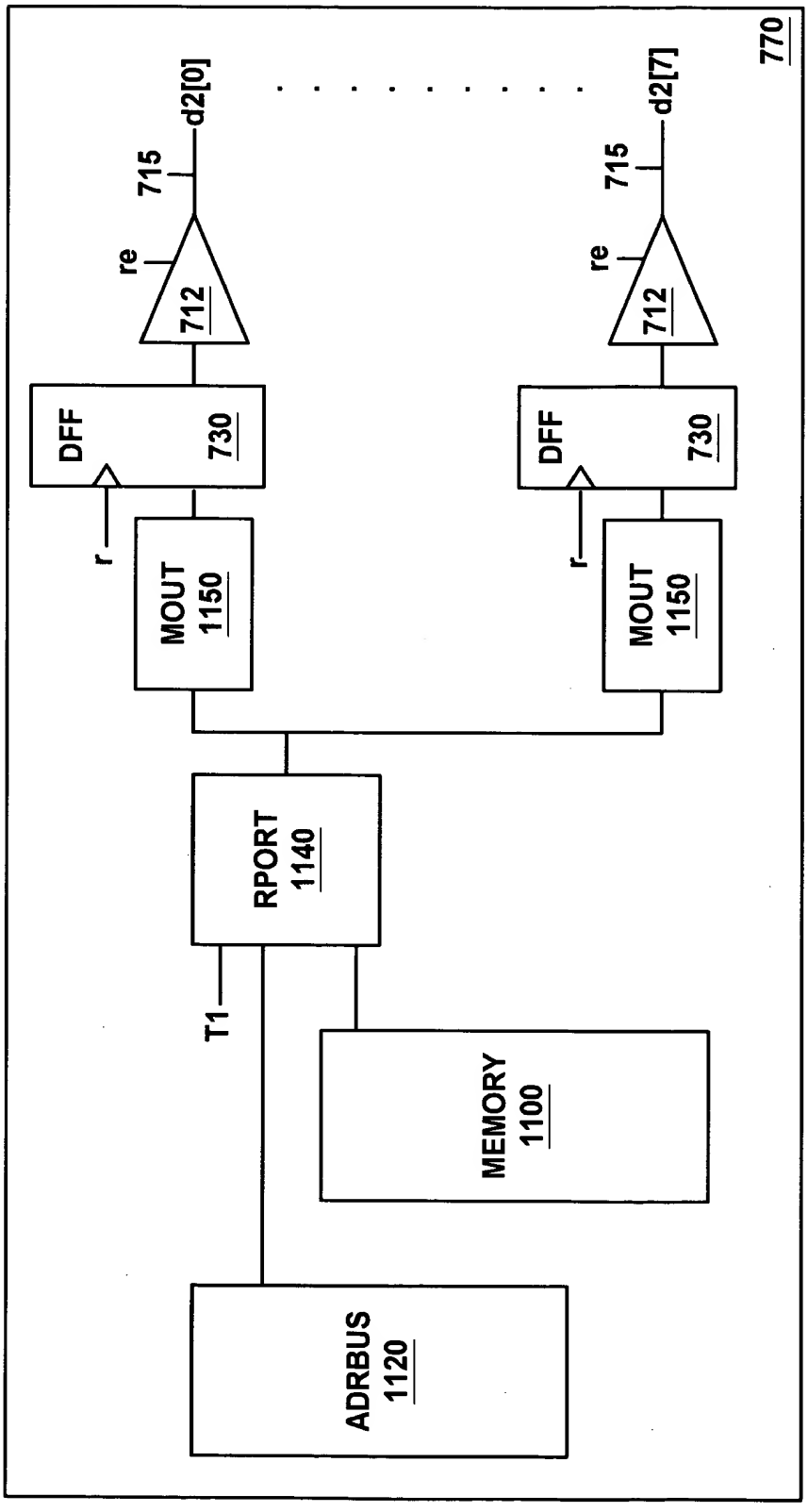
770
d2[0]
d2[7]
715
715
re
re
712
712
DFF
730
DFF
730
r
r
MOUT
1150
MOUT
1150
RPORT
1140
T1
MEMORY
1100
ADRBUS
1120

FIGURE 7C

800

```
module atpgram ( r,w,  a,  d1, d2 );
input  r,w;
input  [127:0]  a;
input  [7:0]  d1;
output  [7:0]  d2;
     reg  [7:0]   mymem  [15:0],   d2;
     integer  i;
function  [6:0]  addr_encode;
input  [127:0]  addr;  //  decoded address
integer n;
begin
      addr_encode = 7'bx;  //  init
      for  (n=0;  n < 128;  n=n+1)  begin
           if  (addr [n] ==1)  begin
               addr [n]  =0;
               if  ( l addr  == 0)  addr_encode=n;
               n = 128;  //  break
          end
       end
end
endfunction

     always  @  (posedge w)  if  (a)
                      mymem [addr_encode (a) ] = d1;
     always @  r if  (r  && a)
                    d2 = mymem [addr_encode (a) ];
endmodule
```

FIGURE 8

```
input [7:0] comp;
output [3:0] addr;
output hit, mhit;
hit = 0; mhit = 0;
addr = 4' bX;
for (i=0; i<16; i=i+1) begin
  if (mymem [i] ==comp) begin
     if (hit==1) mhit = 1;
     hit = 1;
     addr = i; // keep last address
  end
end
```

900

FIGURE 9A

```
input [7:0] comp, mask;
output [3:0] addr;
output hit, mhit;
hit = 0; mhit = 0;
addr = 4' bX;
for (i=0; i<16; i=i+1) begin
  if ( (mymem [i] &mask) ==(comp&mask) )
  begin
      if (hit==1) mhit = 1;
      else addr - i; // keep 1st address
      hit = 1;
  end
end
```

940

FIGURE 9B

```
input [7:0] comp, mask;
output [15:0] addr;
output hit, mhit;
hit = 0; mhit = 0;
addr = 0;
for (i=0; i<16; i=i+1) begin
  if ( (mymem [i] &mask) ==(comp&mask) )
  begin
      if (hit==1) begin
      hit = 0;
      i = 16; // break;
   end
   else addr [i] = 1; // keep 1st addr
   hit = 1;
  end
end
```

970

FIGURE 9C

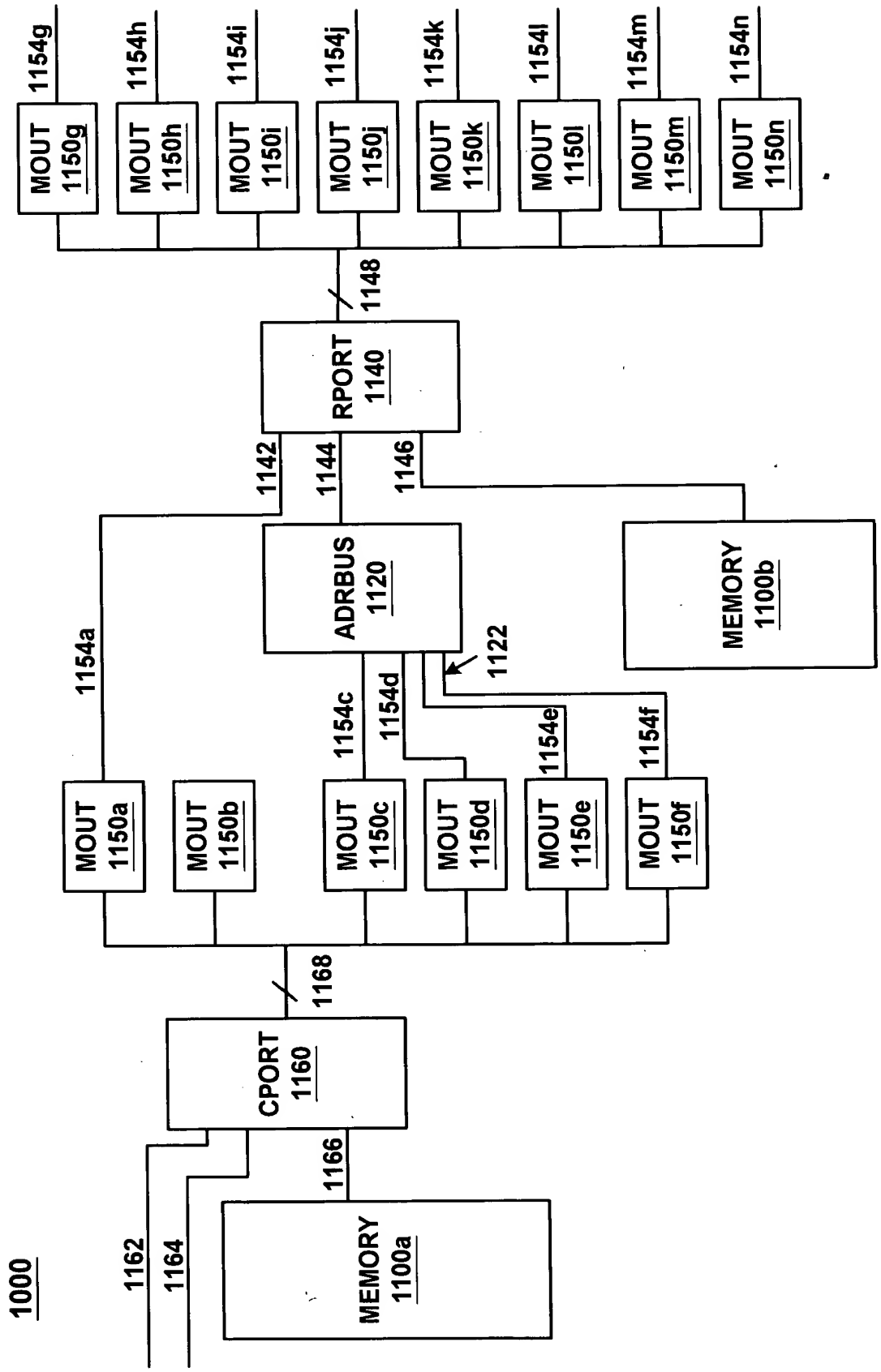
1000
MEMORY 1100a
1162
1164
CPORT 1160
1166
1168
MOUT 1150a
MOUT 1150b
MOUT 1150c
MOUT 1150d
MOUT 1150e
MOUT 1150f
1154a
1154c
1154d
1154e
1154f
ADRBUS 1120
1122
MEMORY 1100b
1142
1144
1146
RPORT 1140
1148
MOUT 1150g
MOUT 1150h
MOUT 1150i
MOUT 1150j
MOUT 1150k
MOUT 1150l
MOUT 1150m
MOUT 1150n
1154g
1154h
1154i
1154j
1154k
1154l
1154m
1154n

FIGURE 10

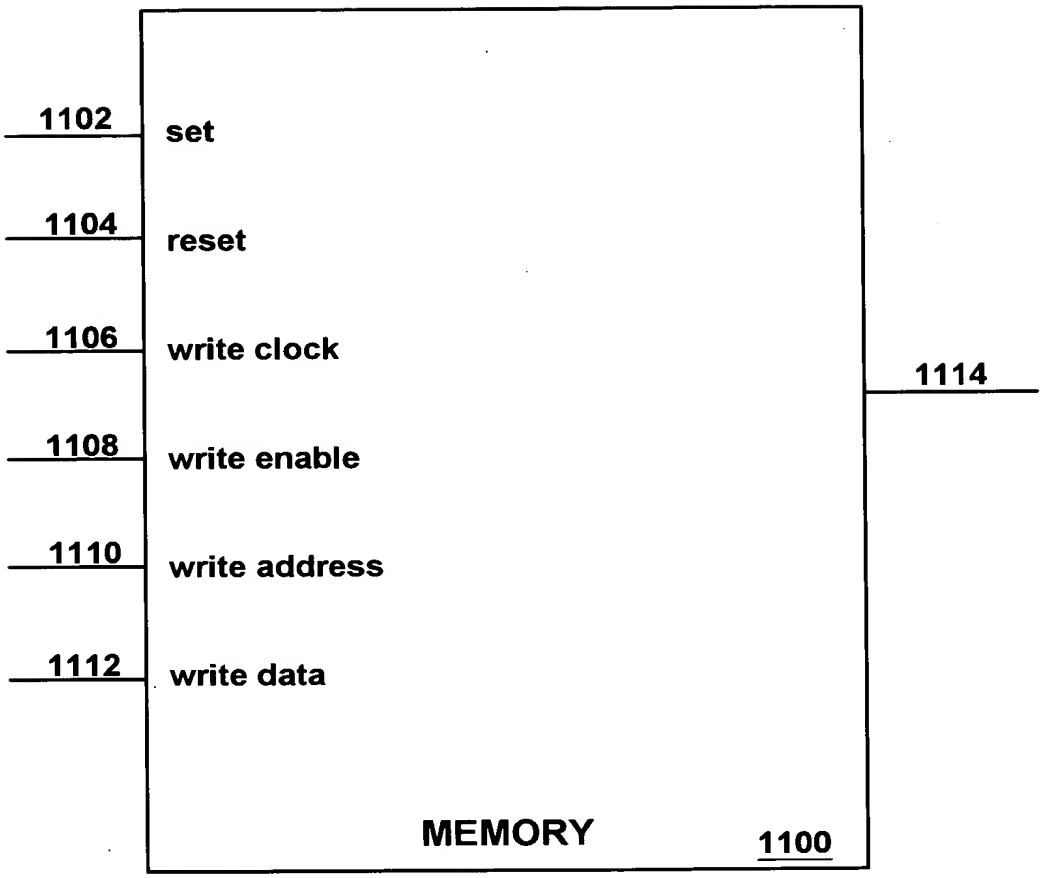
1102
set
1104
reset
1106
write clock
1108
write enable
1110
write address
1112
write data
1114
MEMORY
1100

FIGURE 11A

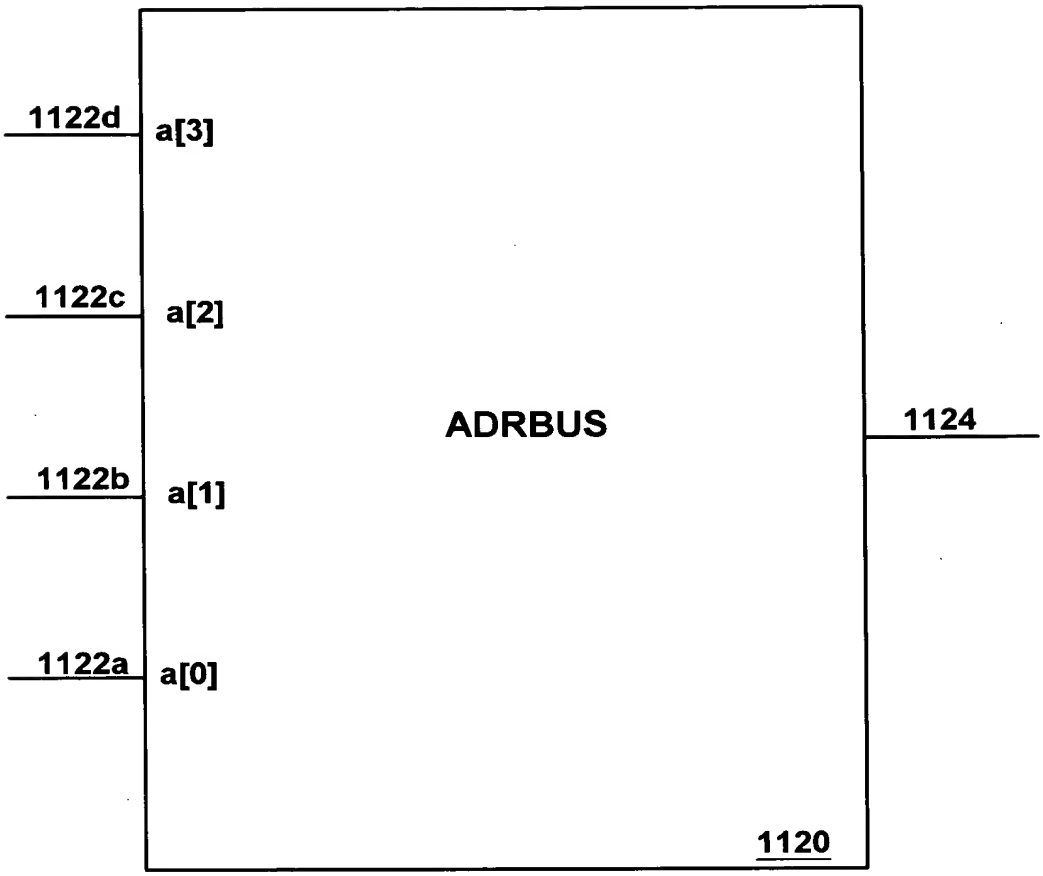

FIGURE 11B

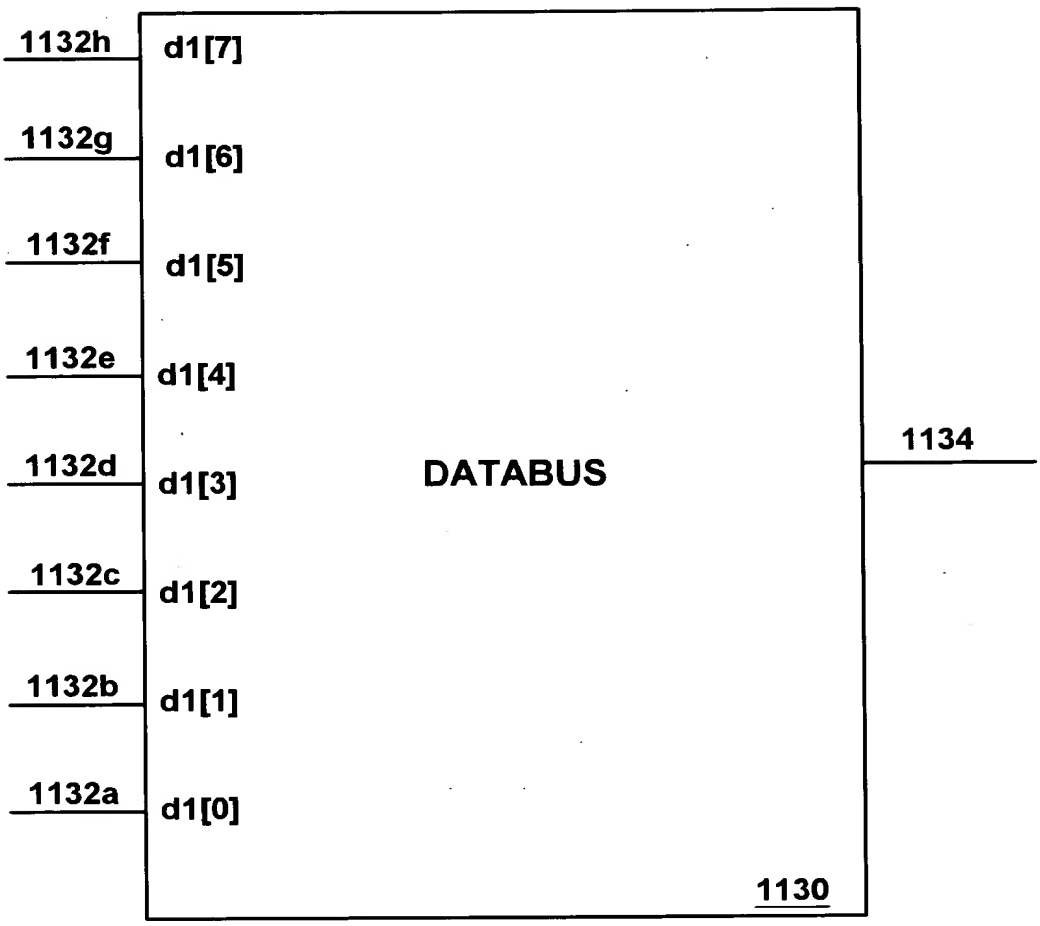
1132h
d1[7]
1132g
d1[6]
1132f
d1[5]
1132e
d1[4]
1132d
d1[3]
1132c
d1[2]
1132b
d1[1]
1132a
d1[0]
DATABUS
1134
1130

FIGURE 11C

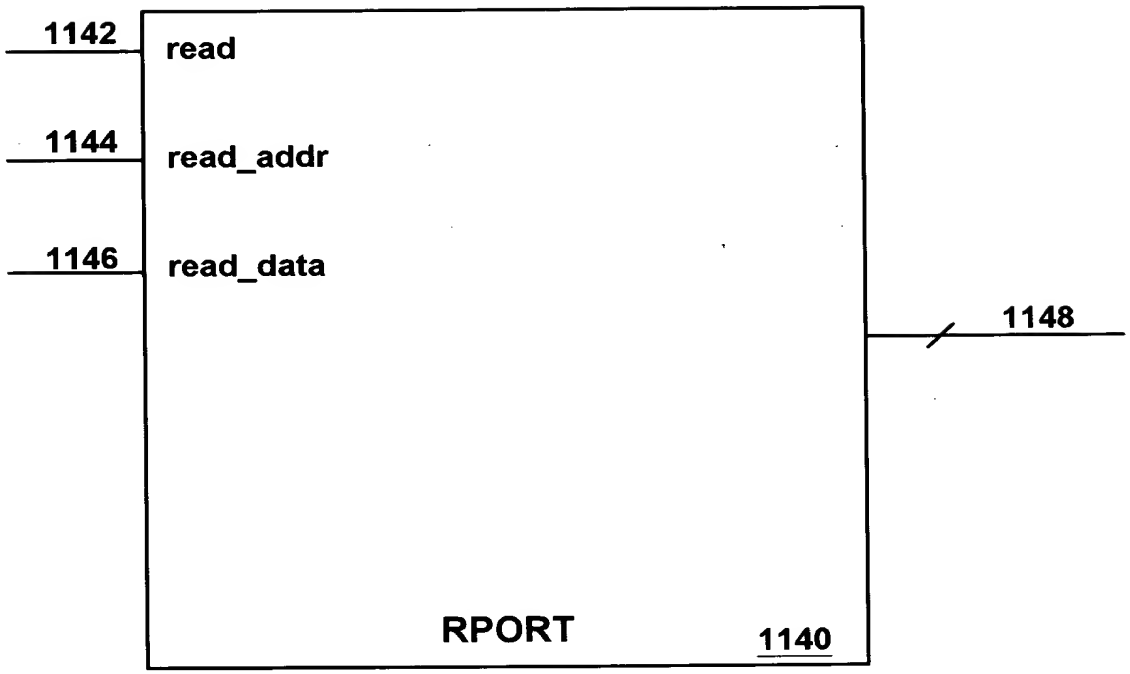

FIGURE 11D

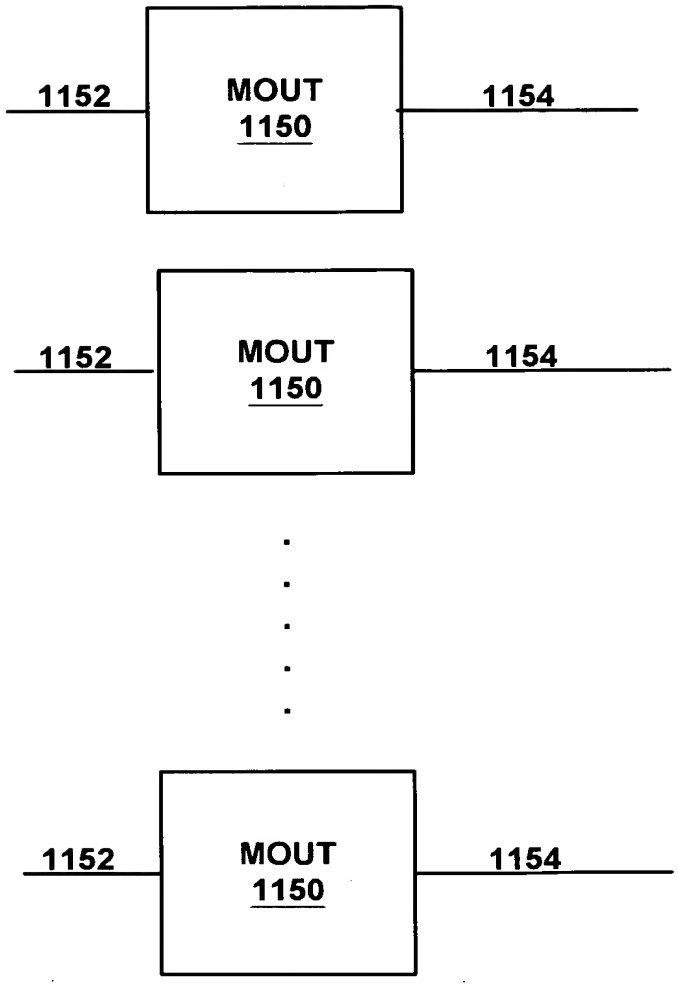

FIGURE 11E

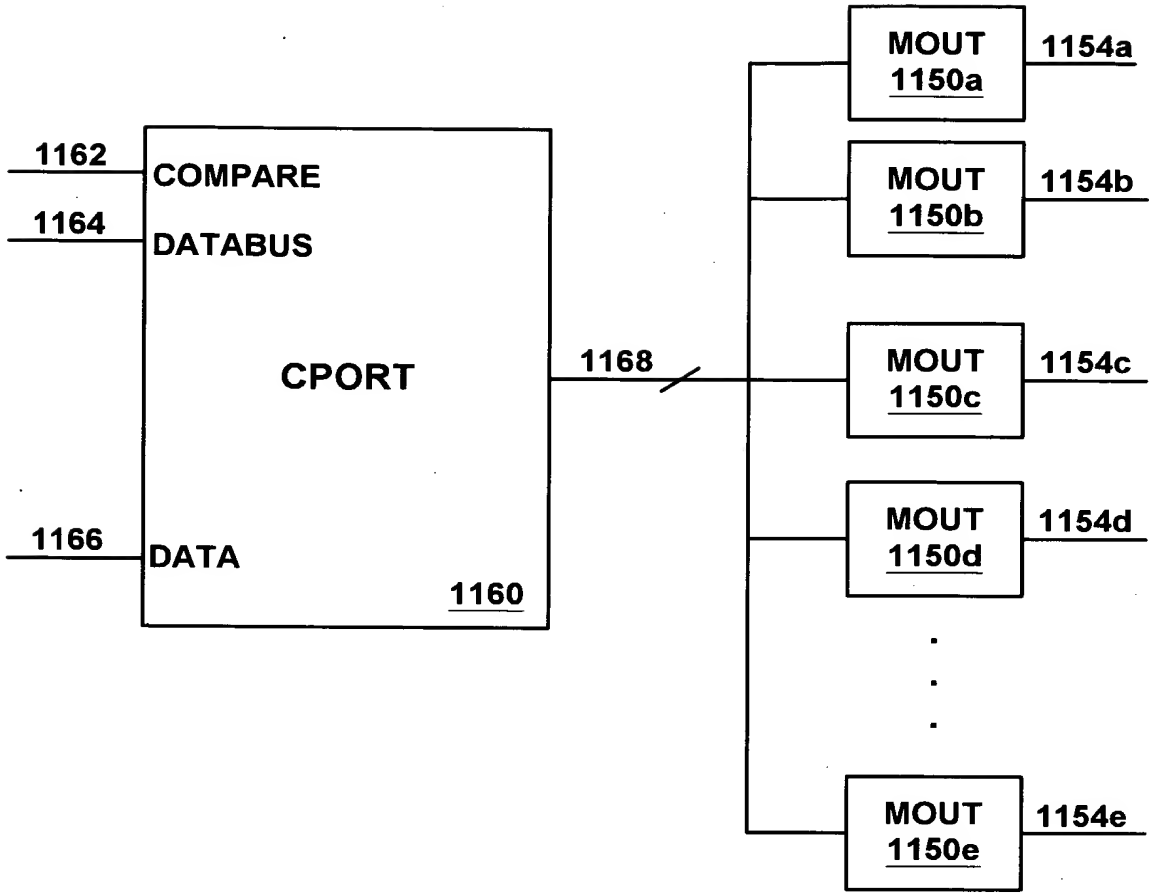
1162
COMPARE
1164
DATABUS
CPORT
1166
DATA
1160
1168
MOUT
1150a
1154a
MOUT
1150b
1154b
MOUT
1150c
1154c
MOUT
1150d
1154d
MOUT
1150e
1154e

FIGURE 11F

```
module withram (reset,  r,w,  a,  d1,d2);
input reset,  r,w;
input  [3:0]  a;
input  [7:0] d1;
output  [7:0]  d2;
     reg  [7:0]  mymem  [15:0],  d2;
     integer  i;
     event mymem_update;
     always  @  reset  if  (reset) begin
     for  (i=0;  i<16;  i=i+1)  mymem [i]  <=0;
     #0;  ->mymem_update; end
     always  @  (posedge w)  begin
     mymem [a]  <= d1;  #0;  ->mymem_update;
     end
     always  @  (r or a or mymem_update)
     if  (r)  d2  <= mymem [a];
    else  d2  <=  8'b0;
endmodule                                    1200
```

FIGURE 12